AFRL-RI-RS-TR-2016-133

PLATFORM FOR HIGH-ASSURANCE CLOUD COMPUTING

CORNELL UNIVERSITY

JUNE 2016

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2016-133 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
JUANITA RILEY
Work Unit Manager

/ S /
WARREN H. DEBANY, JR.
Technical Advisor, Information
Exploitation & Operations Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* JUNE 2016	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* SEP 2011 – DEC 2015

4. TITLE AND SUBTITLE PLATFORM FOR HIGH-ASSURANCE CLOUD COMPUTING	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA8750-11-2-0256
	5c. PROGRAM ELEMENT NUMBER 62303E
6. AUTHOR(S) Ken Birman	**5d. PROJECT NUMBER** MIRCO
	5e. TASK NUMBER CO
	5f. WORK UNIT NUMBER RN
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Cornell University Department of Computer Science Bill & Melinda Gates Hall Ithaca, NY 14853	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIGA 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2016-133

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Cornell's MRC effort undertook to help the military leverage cloud computing (elasticity, better data center utilization, automated management). The core issue is that cloud computing is notoriously weak on consistency, fault-tolerance, and security. With our new cloud computing "tool chain", applications needing high assurance can be implemented with ease and in the same general manner as is used to create today's standard cloud computing applications and services. Additionally, our SuperCloud (a related but distinct project under the same MRC funding) reduces vendor lock-in and permits application to migrate, to follow the sun or leverage inexpensive cloud resources. Jointly, these efforts offer a major advance in the ability of the government and military to leverage the cloud. We demonstrated our work in an area important to the military: the smart power grid, protection of which has been identified as a major national defense priority. At the same time, progress towards a smarter grid offers a chance to save power and increase utilization of renewables. Using our MRC tool chain, we created GridCloud, and worked with ISO NE, NYPA and NY ISO to show that our solutions could host a resilient cloud in support of a smart bulk power grid.

15. SUBJECT TERMS
High assurance distributed computing, cloud computing, smart power grid, hybrid cloud

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** **JUANITA RILEY**
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	U	31	**19b. TELEPHONE NUMBER** (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES

1. SUMMARY

It has been difficult for the US military to leverage cloud computing, for a great number of reasons. Today's commercial cloud and even the government cloud offerings evolved in response to competitive opportunities and pressures. It offers new capabilities: a rent-on-demand computing model, the potential for dramatic elasticity (launching large numbers of applications on short notice at runtime, and for hosting huge amounts of data), tools for data mining in parallel at massive scale and carrying out machine learning and other optimization tasks on huge data sets, tools for graphical knowledge discovery, etc. On the down side, however, the cloud has a security model strongly shaped by commercial security for web purchases using credit cards. The consistency model is shaped by a desire to rapidly respond to queries using cached data, even if that data might be stale. And the list goes on: cloud users worry about the safety of data on the data center disks, even if encrypted. They worry about vendor lock-in, and about the danger that a critical system might be unavailable at time of urgent need because of some form of third-party cloud outage.

Some of these issues are non-technical, and for those problems, research won't help. But the industry is prioritizing hybrid cloud computing support (where the cloud augments traditional owned and locally-hosted systems), government cloud (where some roles normally played by third parties like Amazon are instead played by military personnel), and real-time cloud (where cloud services are explicitly designed to interface with devices in the environment). Such trends are promising: one can imagine selective use of cloud resources in appropriate ways that don't place inappropriate trust upon the cloud, yet leverage its strengths, which include modes of computing such as very rapid elasticity, where one normally runs in a slimmed down mode but when requirements surge, can allocate a lot of computing resources on short notice, perhaps using a preloaded database that could be huge and that has to be spread over huge numbers of machines. In a private setting, such patterns are very expensive; with a shared cloud, the cost drops dramatically.

The key to our work is to leverage cutting edge technology options but also to enhance and secure them, by layering in extra solutions as needed in a cautious way. For example, our system includes a new and very active form of system management that we combine with an easily used way to replicate data (even data being rapidly updated) in a manner that is secure and strongly consistent. With such steps, we are able to compensate for the limitations of the commercial offerings: the standard cloud has a more relaxed style of system management aimed primarily at the three-tier programming model that prevails in commercial cloud uses, and favors weakly consistent replication solutions that are also at greater risk for insider attacks or other forms of data contamination attacks. Thus with a few small steps we greatly enhance usability of the cloud in settings that want to leverage the cloud yet need more.

In fact these are just two of a number of innovations we created. Our HACC effort (Highly Assured Cloud Computing) in the Mission Oriented Resilient Cloud (MRC) program successfully demonstrated that this approach really can work well. Our work reflects a very dynamic process of creating enabling technologies, but then using them to gain experience, then turning around and employing the insights so gained to improve our work or identify further needs. As noted, we did this by working with the community creating the smart power grid, and

built a real cloud platform called GridCloud for capturing and monitoring power grid state at a scale never previously attempted. GridCloud is real and is actually being rolled out now to monitor the bulk power grid for the US Northeast, initially as a research prototype, but with hope that the live system will come to play a major role. Thus our MRC work has actually impacted one of the nation's top priorities: smart grid matters both because of the benefit in terms of use of renewables, but also because a smarter grid can be protected more effectively against terrorists or other forms of manipulation.

It is important to emphasize that GridCloud is just one application among many that could be created using our MRC-created hardened cloud computing tools and solutions. GridCloud was actually simple for us to build: we just specialized the general purpose tools that our MRC effort created, by writing simple applications that "understand" the power grid data formats and the key resources that characterize power grid system state, then added a simple monitoring and program-launch framework that lets our power grid users import applications from their existing High Performance Computing (HPC) clusters and run them in the cloud (not all HPC applications can support such a move, but it turns out that for the power community, luck was with us and the most important applications migrated with ease). Thus in a few steps our MRC solutions enabled us to solve this pressing power grid need.

In fact, GridCloud breaks important and "disruptive" new ground for the bulk power community: the solution can monitor even a national-scale power grid, is scalable, turns out to be very cheap to own and operate, and quite fast, and further that it can withstand many kinds of failures or attacks. But our solutions are far more general and could solve a wide range of such problems. We think that as the cloud user community gains comfort with the idea that the cloud (or at least the hybrid cloud, using a government cloud as the cloud platform itself) can do these things, cloud computing becomes a far more appealing option for highly sensitive use cases, including military ones.

At risk of a very slight digression, we find it interesting to realize that smart grid is often perceived as a pure problem of applying machine learning to the power grid, as if the infrastructure side of the smart grid problem was a solved thing – as if the cloud can just do the job. But what we with GridCloud is that often, the missing enabler isn't purely a new kind of optimization-based demand-response control solution, or some other expert system or machine-learning tool, but rather a whole system that needs to be robust and to have real-time properties from the bottom up. To enable the creation of those smart tools, we have to have a place to run them, and data for them to run on, and have to get the data to the tool, and the answer back into the field, within deadlines. Huge investment has focused on the potential of machine learning… far less on this middle layer. Our work has an unusual impact because it bridges this gap, and thus enables the community building the machine-learning "smarts" to suddenly apply their ideas in a completely new settings, and to do so without recklessly trusting the cloud to have guarantees that the commercial cloud in fact lacks.

Beyond the power community, we have also worked closely with the military to help educate thought leaders and decision makers on the options and limitations of the cloud. For example, one of us (Birman) met frequently with Air Force leadership in the Pentagon during a 3 year period when that team was tasked with developing a cloud strategy. Through this dialog, the office of the Air Force CTO and CIO (at that time, Mr. Gilligan, succeeded by Mr. Tilotson and then Mr. Kent Werner) and the office of the Chief of US Cyber Command (General Lord) benefitted from our insights and used them to shape their own plans and expectations relative to what vendors could, or could not, reasonably be asked to do in this space. We've also maintained a very good and active dialog with Mark Linderman, Pat Hurley and others at AFRL Rome, and with the DARPA team that ultimately produced the RADICS BAA, which is presently open. Our Vsync software library (called Isis2 until recently) has been downloaded 5500 times since 2012, and the numbers are growing steadily. Our SuperCloud software is being used at IBM and elsewhere. Our dialog with the European Air Traffic Control agency has helped them harden core aspects of their new 4 Flight platform. Thus, we are having real impact on important problems.

2. INTRODUCTION

Cornell's MRC effort undertook to help the military leverage cloud computing (elasticity, better data center utilization, automated management). Military and other high assurance applications have previously been denied these benefits because cloud computing is notoriously weak on consistency, fault-tolerance, and security. With our new cloud computing "tool chain", applications needing high assurance can be implemented with ease and in the same general manner as is used to create today's standard cloud computing applications and services. Additionally, our SuperCloud (a related but distinct project under the same MRC funding) reduces vendor lock-in and permits application to migrate, to follow the sun or leverage inexpensive cloud resources. Jointly, these efforts offer a major advance in the ability of the government and military to leverage the cloud.

We demonstrated our work in an area important to the military: the smart power grid. Protection of the power grid has been identified as a major national defense priority. At the same time, progress towards a smarter grid offers a chance to save power and increase utilization of renewables. The big puzzle is that as technology reaches into the grid, the potential attack surface is growing. Our approach has been to argue that yes, we should seize these benefits, but at the same time, should use the MRC-developed resiliency technologies created by our effort to ensure that the new solutions will be hardened against disruption or attack, scalable, and able to leverage powerful cloud-hosted machine intelligence tools, both to operate the grid more effectively, and to rapidly sense and repel attack.

Working with Independent System Operator New England (ISO NE), New York Power Authority (NYPA) and New York Independent Systems Operator (NY ISO) (owners of the bulk electric power grid in the US Northeast) and with domain experts from WSU in Pullman Oregon, we created GridCloud, which showcases our work in a form they can use directly. The ISOs are using GridCloud in preproduction experiments, and while the work is far from finished, we based a DARPA RADICS proposal on it and believe that with a few more years of research and development, we can solve the broad problem and leave the nation with a power grid that can be defended even against major, carefully coordinated, acts of aggression.

3. METHODS, ASSUMPTIONS, and PROCEDURES

No ambitious system can be viewed as a monolithic whole, and this is definitely the case for our work. Instead, we like to think of our project as a hierarchical set of software layers: more foundational ones at the bottom of the stack, and then higher level layers enabled by the basics that reside at higher levels of the stack. The integrated whole enables the ambitious demonstrations we mentioned, such as our GridCloud platform for the smart grid. But within GridCloud one finds high level domain-specific solutions that run over lower level frameworks, which in turn have assurance properties expressed and verified using our formal models and verification methods.

By analogy, one could think of the existing commercial cloud. From the bottom up the cloud starts with virtualization (or perhaps today's Docker style of partial virtualization, but the idea is really the same). Large numbers of virtual machines can be launched on a single multicore computer, and this is the bread and butter of cloud technology. With the VM is an operating system, and it hosts elaborate services, and those in turn support domain-specific applications. So you can see how again, there are layers present. Our software has layers one can understand as extending the VM environment, primarily focused on strongly assured data replication. We add services to the VM environment that make use of our low level layers. And then over these, we offer applications that in turn layer these services. We effectively parallel the standard structure, but add things to it that enhance the normal guarantees.

We'll illustrate this with two examples. As a first one, consider our work on SuperCloud. The SuperCloud idea is to take standard cloud virtualization (Xen, which is widely standard and popular), and extend it to become a "container" that can be moved about. With SuperCloud we can move a cloud image from machine to machine or even from data center to data center. This is technically hard, as explained later in the report, but turns out to be feasible, and it enables a powerful kind of flexibility. For example, the Air Force worries about vendor lock-in: what if cloud resources are obtained from vendor A, but this vendor becomes problematic (which could happen in many imaginable ways). With SuperCloud, everything running on vendor A can be replicated on vendor B or even dynamically moved to B. Thus the military can pull back in situations that cause concerns, and by eliminating this key worry, cloud computing becomes far more feasible. SuperCloud lives in the lowest layer of the cloud: it creates a new kind of virtualized environment but the applications just see a normal version of Linux or Windows hosted in what seems like a normal Xen environment, with no changes needed. As explained later, we even virtualize the computer network and move it with the containers, so that no matter where they are launched, they see what they expect.

Focusing still on the very lowest layer, a second technical centerpiece of our work has been focused on finding highly scalable, elastic, ways to do data replication with strong assurance guarantees. The classical ways of solving high assurance replication problems had not scaled particularly well and such solutions had often been overly disrupted when members join or leave an application. Our approach merged the Paxos state machine replication model with the virtual synchrony dynamic epochs model to factor out expensive membership overheads, taking them off the critical path and enabling ultra-fast replicated data updates and reads in a very strong consistency model. This in turn permitted us to offer a plethora of other services that run over the basic infrastructure, such as a scalable way of managing key-value storage with strong assurance properties.

This first accomplishment allows us to climb the cloud technical stack, by offering strong assurance at higher and higher levels. As noted, we ultimately tackled all levels of the stack: bottom right up to the application itself. Our team worked with theoretical tools and in fact leveraged technology from the DARPA-supported CRASH program, which used automated logic (theorem provers) to harden the same sorts of replication protocols of interest in our effort. We then implemented the best protocols in a cloud-oriented platform, currently available in a beta release from Vsync.codeplex.com.

The Vsync software platform, which was initially called Isis2 and then renamed in 2015, is a software library that makes it very easy to harden applications. For example, suppose that a developer has designed a service that takes images of faces as input and checks rapidly against an evolving database of known threats to see if there is a match. It is much easier to build such a service "statically" than to make it fault-tolerant and scalable. The static version just requires a sample database of images and a high quality face comparison method. In contrast, the fault tolerant version might need to be spread over many machines, to accept distributed updates in real-time, to be secured against disruption by intruders, etc. With Vsync, simple library calls solve these kinds of problems: we can form a "group" of servers that have a shared state, and each can update the group state using simple methods that guarantee consistency, for example by coordinating the order of updates so that conflicting actions are performed in the same order at all copies, and so that any failure is handled consistently by all participants. The original static version of the service is thus transformed into a replicated one. With a further step, our CloudMake software manager can be programmed to automatically restart failed elements, to monitor for load imbalances, or to perform automated upgrades as new versions become available. Thus step by step the hard parts of the job are standardized and automated. It becomes easy to create a military-strength version of our face recognition software, and to scale it to handle huge numbers of clients.

In our approach, Vsync is thus a core component. It lives in the standard cloud and is just a library, coded in the standard way, but this particular library can be used by all sorts of application developers who have needs that come down to data replication, fault handling, and parallel processing. Over the course of our DARPA-funded effort we used a cycle of theory, implementation and experimentation to understand the limits on our own and other best of breed solutions, to build new approaches, and to compare them to the best existing options under serious cloud deployment and load cases. To ensure that our work is valid we teamed with early adopters from research labs at MITRE, LLNL, Department of Energy (DOE) Smart Power Grid, and the Air Force and looked at ways of building real solutions that offer high assurance and yet benefit from the cloud. Over time, additional users surfaced, such as the consortium of smart grid operators we mentioned earlier: jointly, ISO NE, NY ISO and NYPA control the power grid for the entire US Northeast.

We worked to extend the current commercial cloud in specific areas where the commercial cloud has weak assurance properties. These include cloud replication tools (which are slow and lack assurance guarantees), cloud security (which focuses mostly on the kind of security needed for the https interactions used to make purchases), cloud control of internet sessions (TCP is often cited as a weak link between the cloud client and the cloud service) and even the authentication and authorization methods used by cloud services to deal with policy enforcement. By importing strong models that have a rigorous theoretical basis and are even supported by tools of the kind being created by our colleagues in the DARPA CRASH program we are able to offer libraries that run on standard cloud platforms but can do far more than those standard platforms normally permit, notably in respects concerned with data replication, consistency, coordination and parallel processing. With our work on GridCloud we went beyond the original DARPA MRC goals and were even able to demonstrate highly assured embedded sensing and control solutions that reside on the cloud but can monitor an infrastructure that might be widely distributed and massive in scope.

Our deliverables focused primarily on new theoretical models and results, implementation of those results in the Vsync platform, and papers reporting on our challenges dealing with massive scale, rapid elasticity events and other events seen in cloud settings (which often experience transient overloads, bursts of packet loss, and bursts of failures, to list just a few such challenges). The resulting deliverables thus consisted of research papers and open-source software distributions, as part of the evolving Vsync platform (Vsync.codeplex.com). Examples of papers we created include [1,2,3,4,5,6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20,21,22,23,24,25 ,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44].

New versions have been made available for all our software deliverables from time to time, sometimes as often as every few months, sometimes less frequently, fixing bugs and incorporating our latest advances. Demonstration programs and applications were released, as well as documentation and self-install scripts. We also improved Birman's textbook [3], transforming it into a how-to guide to building reliable distributed cloud computing solutions. Springer-Verlag released the new edition in early 2012.

Our open source approach to technology release has made it easy for potential commercial or military users to work with Vsync and other Cornell developed technologies. We also provide additional help in some cases, including consulting via phone or in person, short courses, etc. We are not currently planning a commercial spin-off for Vsync (this could change but currently seems unnecessary); our view is that there is a strong demand for high assurance cloud computing, but that any required technologies need to be as inexpensive as possible to match the free technology offerings from the major vendors.

4. RESULTS AND DISCUSSION

In this section we give a tool by tool summary of the approach we tool to our main subsystems and software components. Each corresponds to far more extensive discussion in papers, technical reports and documentation, hence we treat this section as a review but not a deep dive: a deep dive would require a text-book length report. But we do encourage the interested reader to turn to our definitive papers for the details we elided here.

SuperCloud: Our approach and accomplishments

Supported by DARPA MRC, the Supercloud platform is a nested virtualization platform, running Xen as a virtual machine monitor (VMM) "within" a normal cloud virtualization infrastructure and using a virtualized software-defined network (SDN) overlay to encapsulate the desired network environment [45,42,46,47] Applications running in the Supercloud environment can be transparently and automatically mirrored at multiple locations, migrated live from place to place, frozen (and migrated), or co-hosted with several instances side by side on the same machine, all with no changes to the software running within the Supercloud itself, which sees a full Linux or Windows environment and a full enterprise network stack, secured using cutting-edge SDN techniques.

Supercloud has been deployed using resources from several major cloud providers, including Amazon Elastic Compute Cloud (EC2), Rackspace, Hewlett-Packard (HP) Cloud, and some private clouds. We have developed a new storage system for the Supercloud in which we decoupled consistency management from data transfer. In the data transfer module, we can implement various schedulers for transfer. On-demand transfer can lead to large delays but it is free while a VM runs. Using a "hot-standby" policy all dirty pages are immediately copied to other locations, which reduces downtime during migration but it is exceedingly expensive while a VM runs. We offer a "best of both worlds" option that tries to predict which pages are not likely to be modified again and likely to be loaded after migration. We have also developed various schedulers for VM placement and migration itself. We can for example leverage the Amazon EC2 spot market and get significant savings from smart placement.

A second line of work yielded a compelling response to those who worry about cloud vendor lock-in and other limitations. This work was done in part jointly with IBM and is already shaping IBM's product offerings in the space. It represents a breakthrough that eliminates the kinds of problems that previously inhibited military and government use of "public" clouds, to the extent that AFRL is currently working with us to apply the prototype in some real Air Force scenarios of interest. The Supercloud provides virtual storage that spans the various underlying clouds

Our contributions are in various dimensions:

- We developed the first Supercloud that spans heterogeneous clouds, often under various administrative domains, and showed how this can be done efficiently.
- We developed and demonstrated various use cases for application-level migration as a first-class primitive. (Other clouds implement VM migration but do not expose it to applications).
- We developed novel scheduling and placement algorithms for heterogeneous clouds.

- We were among the first to apply Software-Defined Networking technology spanning multiple network domains.
- We developed the first geodistributed storage system specialized for VM images.

We believe that our technology can have a disruptive effect on the way companies use clouds. A major trend in company cloud use are so-called *Hybrid Clouds*: a Hybrid Cloud is a cloud that consists of resources in a company owned cluster, backed up by a public cloud. When the company experiences a load surge, it can utilize the public resources to add capacity. If the company needs to test new software, it can utilize the public resources without affecting the private resources. While extremely popular, it requires that the private cloud run *identical* cloud software as the public cloud. In practice, this means that companies that want to use a Hybrid Cloud have to use VMware virtual machines and use the VMware public cloud for backup. They cannot use open source virtual machine monitors and they cannot use popular cloud providers such as Amazon EC2, Google Compute Engine, Microsoft Azure, and so on (although some of these are starting to come out with their own proprietary solutions).

Our Supercloud technology can completely change all this. We already demonstrated virtual machines migrating between various open source cloud software for private clouds and various public cloud providers. We should thus be able to significantly change the Hybrid Cloud landscape.
The Supercloud supports *Smart Spot Instances*: Instead of making a trade-off between cost and availability, Smart Spot Instances can achieve low cost and high availability at the same time with the support of user-level live migration. User VMs runs as second layer VMs (*sVMs*), while the Amazon spot instances form the first layer VMs (*fVMs*). Then sVMs can be migrated according to a specified scheduling policy.

Smart Spot Instances can increase availability compared to ordinary spot instances. Users set a high maximum bid, which can be as high as the on-demand prices. The Supercloud monitors the spot price and live migrates sVMs to other, possibly cheaper fVMs – either just another spot instance type, or even spot instances in another availability zone. In the worst case, the sVMs can be live migrated to normal on-demand instances that are always available, so the user only needs to pay as much as the on-demand price. When the price approaches the maximum bid in the middle of an instance-hour, the Supercloud migrates the sVMs to avoid being terminated.

Smart Spot Instances can greatly reduce costs by taking advantage of price diversity in different types of the spot instances. When the end of an instance-hour approaches, the Supercloud compares the spot prices in each spot instance types and their regular prices, and live migrate sVMs to the fVMs that can achieve lowest overall costs regarding the different capacity of different instance types.

A Supercloud allows companies, organizations, and individuals to move to a cloud computing environment while retaining control over placement and scheduling. In particular, a cloud user controls the location and live migration of their computation, networking, and storage without owning all of the underlying infrastructure—a level of control that is not available today.

The Supercloud can take advantage of spot market pricing. It can migrate spot market instances across availability zones and providers, significantly reducing the cost for services and applications operated in the cloud. Further, we showed that the Supercloud provides significantly decreased and more predictable latencies than any of the underlying cloud providers individually. We also described the current status of the Supercloud implementation and discussed ongoing challenges.

SuperCloud is our virtualized environment for cloud computing with transparent and automated migration from data center to data center even across vendors, using our standardized VM container. At the start of this particular subtask, we had a running SuperCloud prototype, but it was not able to minimize the end-user costs. For example, SuperCloud treated the client system as if it was just a single VM instance, when in reality a cloud client often runs a virtual cluster, so that in fact a group of VMs should be scheduled. We set out to do a very systematic study of how to best exploit the Amazon Spot Market in the face of rational behavior (in the game theoretic sense).

Without getting overly technical, SuperCloud is composed of three main parts. One is a kind of "wrapper" to let it run on whatever the VMM on the cloud vendor system happens to be. Different vendors use different core infrastructures, and the "XenBlanket", which is our name for this element, adapts the remainder of the system to match. The second component looks like a standard Xen VMM, but in fact is adapted to support state snapshots and migration and to match the API of the XenBlanket. These small changes ensure that SuperCloud can run on any cloud vendor system and can snapshot its state, ship it to another machine or even another data center, and restart there. The third component is a virtualized network based on the OpenSDN model from OpenStack. With these three running in concert, SuperCloud is able to support some striking new functions:

- We can migrate a running application with low overheads, sending it to another machine locally or to one located (very) remotely.
- We can follow the sun: a series of migrations aimed at keeping a server close to the most active group of users (imagine here a kind of server that sees peak use from 9am-10am local time).
- We can follow cloud "spot pricing" and migrate to the cheapest resources, continuously.
- We can move from a damaged data center or one under attack to a safer one.
- If a system has some resources that must be hosted locally and cannot be used from the cloud or moved into the cloud, we can migrate the application to those resources (like a database of images, a special radar device, etc.) as needed, use the resource briefly, then migrate back to the cloud once the activity is finished.

Much of our SuperCloud work was done jointly with IBM, which is developing commercial spinoff products. Thus this research will rapidly transition into commercial practice.

Vsync: Scalable and Consistent Reliability for the Cloud

Vsync is a powerful new option for cloud computing that can enable scalable, reliable, secure replication of data, coordination and fault-tolerance, and can guarantee data consistency at high update rates even in the highly elastic first-tier of the cloud. Vsync is actually a recent name: our work started out using the name Isis2, but we changed it in protest after the terrorist attack in late 2015 in Paris.

The new name is a reference to the formal model used by the system, namely *virtual synchrony* [3, 48]. The model is a form of state machine replication with various optimizations available (but optional) that permit greater speed without loss of correctness. This final report is not an ideal setting for showing code samples, but we have created many hours of videos to teach people how to use the system, demo programs to show how it works and that they can transform into versions solving their own needs, and even posted some recommended projects for people who are learning and need pointers on what they might try to develop to get a feel for the system. Developers work in C#, IronPython, IronRuby, F#, C++/CLI or any of the dozens of other programming languages available for Microsoft's .NET framework, or on Linux using the Mono compiler and runtime libraries. Vsync even automates use of cutting edge new communication tools such as Remote Direct Memory Access (RDMA) zero-copy data movement from one machine to another, using a reliable memory-to-memory DMA transmission at optical line rates.

Extensive materials and teaching tools on the system can be found at Vsync.codeplex.com [5]. In summary, the Vsync software library helps developers build applications that will run on multiple computers, coordinating actions, sharing replicated data, moving files and other information at high speeds, cooperating to support key-value storage (DHT storage), etc. Vsync aims at sophisticated developers with challenging needs, and is designed to be highly secure, fault-tolerant, consistent and very scalable, even under "cloudy conditions." For example, due to virtualization and multi-tenancy, cloud nodes often become overloaded for brief periods and hence show long scheduling pauses, and they sometimes crash in unexpected ways at rates that might seem surprising in other settings (these are actually fake crashes caused when the management layer decides to shut a node down and perhaps start some other image on some other node to replace it). Notice that with SuperCloud, such an application could be continuously mirrored or, if mirroring is impractical, perhaps dynamically migrated. But if the update rates are high and the overheads of SuperCloud are infeasible, sometimes we just build special purpose solutions, and Vsync aims at this case.

These are complex systems to interoperate with and reliable, consistent, secure data replication is hard to pull off, particularly in a strong and theoretically sound model. The premise behind this part of our project is that the need for such solutions is rapidly growing and hence that if we can package the needed solutions, developers will want to use them: with the trends towards data centers of all sizes and shapes (ranging from small racks of just a dozen or two machines to massive cloud computing data centers with hundreds of thousands of them), developers of modern computing systems need to target the Web, employ Web Services APIs, and yet somehow ensure that the solutions they build can scale out without loss of assurance properties such as data security (who knows what the other users of the cloud might be doing… or what might be watching?), consistency and fault-tolerance. The Vsync library was built to help developers solve this problem in an easy way, closely matched to the style of development used for standard object-oriented applications that use GUI builders.

After four years of active use by a diversity of developers, Vsync is quite stable now and rather mature. Meanwhile, we have begun some basic follow-on work, also sponsored by DARPA MRC, but under a reduced burn-rate extension after our main funding ended.

In the follow-on activity we created a basic and very simple reliable data replication layer, Reliable DMA Multicast (RDMC), which runs on RDMA and offers reliable multicast using zero-copy RDMA, by building a highly efficient overlay tree and sending data in chunks, using reliable one-to-one RDMA transfers. A paper on this work was sent to NSDI, not accepted, and is now being revised for submission to Middleware, but the software itself is working and offers the *world's fasted reliable data replication solution, whether for one copy or hundreds.* RDMC is coded in C++ and easy to use. The download site is rdmc.codeplex.com [8, 49].

A second paper, just submitted to SIGCOMM, details another RDMA solution: the Shared State Table, or SST. SST is a framework with which nodes in a single rack connected by RDMA can share local states and detect system events. An SST over a group of nodes is a table consisting of a row for each member of the group and columns representing state variables. We provide a mechanism for defining system events (called predicates) and callbacks (triggers) over the entries of the state table. The triggers are executed when the predicates are detected to be true. SST focuses on minimizing the time to detect predicates with the secondary concern of judicious use of RDMA resources. SST optimizes for RDMA operations and abstracts them from the programmer, thus providing a convenient interface to code applications to run over RDMA networks. Our preliminary work carefully examines the characteristics of one-sided RDMA reads and writes to support this model, arrives at an optimal design for the SST, and provides new insights into system design using RDMA for peer-to-peer architectures. SST adds minimal overhead over raw RDMA primitives and scales linearly with the number of nodes. Our experimental study measures delay for detecting different categories of predicates (we can reach event rates of 500K per second or more), and illustrates the use of SST in a scenario based on OSPF routing. Again, the plan is to offer an open source release shortly [44].

This work will allow us to create a next generation Vsync solution that should run at data rates 1000x or even 10,000x faster than Vsync or any other of today's replication technologies. By doing so we enable huge speedups across the spectrum: in the low levels, but also in file systems, in MapReduce computations, in real-time monitoring and management systems, etc. We anticipate a revolutionary advance in decision-making tools, all made possible by blindingly fast data replication, enabling instant elasticity: something happens, and thousands of tasks can be assigned to tackle it, even if the data input to those will be large.

IronStack: A secure and resilient network management infrastructure
IronStack is a resilient network management solution aimed at a communication network that connects devices residing outside the cloud, such as sensors or video cameras, back to servers in the cloud-hosted data center. Some settings have in-house solutions, and would not need IronStack, but for developers faced with creating such a system or desiring to upgrade a balky and idiosyncratic one, IronStack could be a good option. Developer Z. Teo has focused on creating an open source preliminary version for use at Cornell and as a preproduction demonstration, but is planning to spin off a commercialized version soon. The company will eventually offer a turn-key resilient network, 24x7 support and deployment help.

IronStack borrows some ideas from redundant array of independent disks (RAID) [50], a set of redundancy schemes commonly used to protect data by utilizing multiple hard drives. Analogously, the redundancy in data networks are provided by multiple disjoint paths from a source to a destination. However, current networks do not usually feature multiple disjoint paths because they are tedious to design, require a fair degree of manual configuration, and are difficult to maintain in a safe configuration over extended periods of time [51]. Also, software that takes advantage of multiple paths is rare in practice [3]. IronStack solves these problems by automatically generating a safe configuration for any given network topology, while allowing multiple paths to be used simultaneously without the need for any laborious configuration or forethought. IronStack is self-configuring, self-adapting and self-healing, so repeated changes to the network topology do not affect the operation of dependent network software. Thus, IronStack automatically manages the network efficiently in a way that is transparent to users.

The way IronStack uses multiple paths in the network can be seen as a continuum of tradeoffs between latency/reliability and bandwidth efficiency. At one extreme end of the spectrum, each packet in a flow can be replicated onto multiple disjoint paths. The receiving end delivers the first arriving packet to the application and discards the duplicates. Such a scheme minimizes latency and improves the stability of the flow, while also tolerating up to $n - 1$ link or switch failures, at a cost of n times the bandwidth. On the other extreme end of the spectrum, each disjoint path can be seen as a separate channel through which flows can be sent through, so each successive packet in a flow can be sent down whichever path is first available (thus avoiding the problem of sending too many packets down congested paths). In a lightly loaded network, approximately $1/n$ of the packets in a flow can be sent down each path. This scheme maximizes bandwidth efficiency but clearly sacrifices on flow stability and latency, since the entire flow is now dependent on the slowest link. It also does not tolerate link failures although such tolerance may not be necessary if the software protocol can handle it (e.g. TCP with selective acknowledgements).

The IronStack controller is capable of limited data transfers between a switch's control plane and its data plane. It is thus possible for IronStack to entirely handle packet processing on behalf of the end hosts. The advantage of such an approach is that it enables all IronStack capabilities to the end hosts without necessitating any kind of hardware or software change, with the effect that end hosts are completely unaware of an underlying change to the network. In light of the difficulties in modifying individual network equipment operating system kernels as mentioned in the preceding section, such transparency is valuable since changes are localized to the switching equipment and its controller. However, this approach is not scalable because the bandwidth for transfers between the data plane and the control plane 4 is limited. Consequently, it is not possible to service too many concurrent IronStack function requests simultaneously.

In work still underway, IronStack will be leveraging Vsync to replicate its controller using a form of state machine replication that Vsync offers as a kind of turn-key API. This will make IronStack itself highly robust against disruption, resulting in a complete self-managed security and reliability story for owned networks that need to interface to cloud-hosted systems.

IronStack has been used to operate Cornell's Gates Hall network, which is a deployment of about

12 SDN switches and routers playing a complex mix of experimental and production roles. Interesting, Cornell turned to IronStack in frustration when commercial options like OpenDaylight and the NEC product simply didn't work, and IronStack was an instant success. By now Cornell has moved most of the production work off IronStack simply because having a graduate student run our building raises awkward security concerns, but the technology performed flawlessly for six months and continues to run our experimental SDN deployment, with the exception of a few nodes running a competing SDN solution by faculty member Nate Foster. IronStack publications focus on its automated methods for secure connection management, robust traffic splitting or replication over multiple routes, and its leveraging of NetFPGA hardware for packet encoding and deduplication. An additional paper reports on experience with the IronStack deployment in Cornell's Gates Hall.

CloudMake

Earlier we gave an example of transforming a face-recognition program developed "statically" into a version that can run as a resilient service in a cloud. We mentioned that one task is to monitor the health of the service and restart elements of it as needed, when needed, under policy control by the developer. This isn't trivial because some services depend on others: perhaps before we can launch the server on a given node, we need to be sure that the image retrieval service is up and running on that node, and perhaps this in turn depends on running a copy of Oracle or SQL server or MySQL.

CloudMake is a dependency specialist. We based it on the popular Linux Make utility, which represents dependencies between code modules: if you modify app.h, and app is compiled from app.h and app.cpp, then Make is used to rebuild app, and anything depending on app.h too. We realized that by writing down cloud management events (loads, node starts and stops, etc.) in small files, then when a program halts, its status file will update to reflect this, and we could then run a version of Make that would notice the failure, realize that the system configuration depends on the health status, recompute a new configuration, and then this can trigger launching a new copy, setting off an alarm, etc. CloudMake does all of these things. It uses a plug-in optimization module: a so-called "constraint solver" that can actually be any of a few dozen such systems, and will automatically do the mapping of application components to available resources. Graduate student Theo Gkountouvas used Vsync within CloudMake, and hence was able to completely avoid duplicating the needed logic to make CloudMake itself resilient, scalable and fast. CloudMake and GridCloud papers include [1, 10, GA+14, GBA14, ML+13, JB14]

Freeze Frame File System

We built the Freeze-Frame File System (FFFS) over our basic tool chain to demonstrate that with our tools, it is possible to capture real-time data and then provide secure, strongly consistent real-time mirrored data sharing. The name evokes the image of a film strip, and this intuition is appropriate because FFFS offers a novel real-time snapshot capability. One can do similar things with modern event driven database systems, but for data that isn't naturally structured as database records, simple files (append-only logs or updated tables) are often more natural. FFFS offers the same basic API as any file system, but can do reads at any desired instant in time. Further, it uses RDMA to provide zero-copy data reads and writes.

FFFS allows applications to explore the evolution of the power grid network state at any desired temporal resolution: in contrast to standard parallel computing tools that take a data set from some single instant in time, and apply large numbers of CPUs to carry out a computation on that single data set. Indeed, with FFFS we can also parallelize by spawning a set of tasks that access the power grid state at various points in time, permitting analysis of trends that evolved over a period of time, and we can leverage the massive on-demand parallelism of the cloud. The FFFS system also handles data replication, maintains a remote backup, and is designed to detect any tampering, so that the past state can be used as a trustworthy record of precisely how the power grid state evolved over time. Again, by using Vsync to mirror its own core state and replicating the data stored by "data nodes", FFFS can be strongly consistent and robust against disruptive failures.

GridCloud

Our story comes together in GridCloud, which is the system we created (with some additional funding from DOE) and delivered to ISO NE, NYPA and NY ISO for use in monitoring the smart power grid at the so-called "bulk" level. GridCloud showcases many elements of our DARPA MRC-created technology base, although at this stage does not use SuperCloud. But most of the remainder of our story is present, as the following summary illustrates. In effect, GridCloud is not "the" application, but rather is the first of what we hope will be many solutions to nationally important mission-critical computing tasks that could not have safely run on the cloud with standard commercial software alone, but that can run on a cloud augmented by our tools.

GridCloud has three main elements: a *data capture layer*; an *archival storage* "historian" and a *runtime infrastructure* that operates the system's own components and also launches and supervises power systems analytic applications. VPN and "private cloud" compute-node technology, encryption of data stored into files, and TCP encryption secure the solution against attack.

GridCloud's outward-facing *data collector* applications capture data from the grid, store it into the historian, and then *forward* the data to applications running on the platform. These processes are both numerous but very lightweight: they can run on very inexpensive cloud components. However, inexpensive cloud components are less reliable than more expensive cloud virtual machines and the connections themselves are delay-prone, so we replicate the entire data collection infrastructure. As seen in Figure 1, we aim for 3 replicas per data collector. If failures temporarily knock the number down GridCloud automatically repairs itself. In GridCloud, keys are stored only in the ISO's private key repository. To enable cloud security it is necessary to provide these keys to the cloud infrastructure software, but they would never be stored in persistent cloud storage.

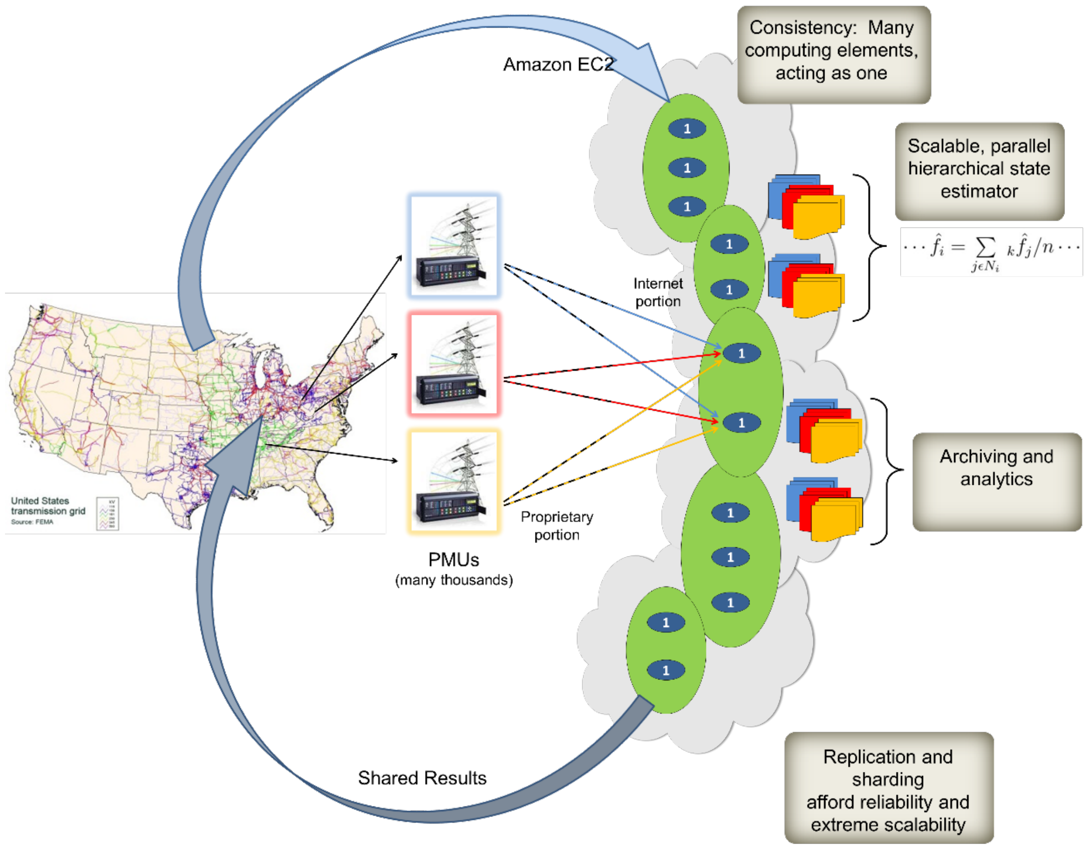
Amazon EC2
Consistency: Many computing elements, acting as one
Scalable, parallel hierarchical state estimator
$\cdots \hat{f}_i = \sum_{j \in N_i} {}_k\hat{f}_j/n \cdots$
Internet portion
Archiving and analytics
United States transmission grid
Source: FEMA
Proprietary portion
PMUs
(many thousands)
Shared Results
Replication and sharding afford reliability and extreme scalability

Figure 1: Overall Grid Cloud Design

Managing a complex cloud system demands continuous observation, adaptation, and fault-tolerance, and one of our early realizations was that existing cloud tools don't fit this need very well: there are many such tools, but they tend to assume that the cloud applications at the edge run in relative isolation, whereas ours is a world of highly interconnected services that talk to one-another. For such componentized systems to work correctly, the key elements must all be running, and if some node shuts down, the applications that were running on it must be relaunched elsewhere. We address this role with CloudMake, the system mentioned earlier. The general architecture can be seen in Figure 2. In steps (1) and (2a,b), CloudMake queries the Operating System (OS) to track application health, while also monitoring application-generated status reports that are stored into XML files (similar to small web pages). If a change in the state occurs (either from the OS or the application processes), the local CloudMake Daemon forwards the change to the CloudMake Leader (3) where the policies are enforced (4). Then, the CloudMake Leader returns all the changes in the configuration to the corresponding local CloudMake Daemons (5) and the configuration is shared between them and the application process in an XML file (6,7). Each policy has an associated leader, which behaves like a single logical entity, but the leader actually runs as a Replicated State Machine to provide fault tolerance, using the Vsync group communication system to implement this form of fault-tolerance. If a component fails, CloudMake immediately launches a new instance to restore full functionality.

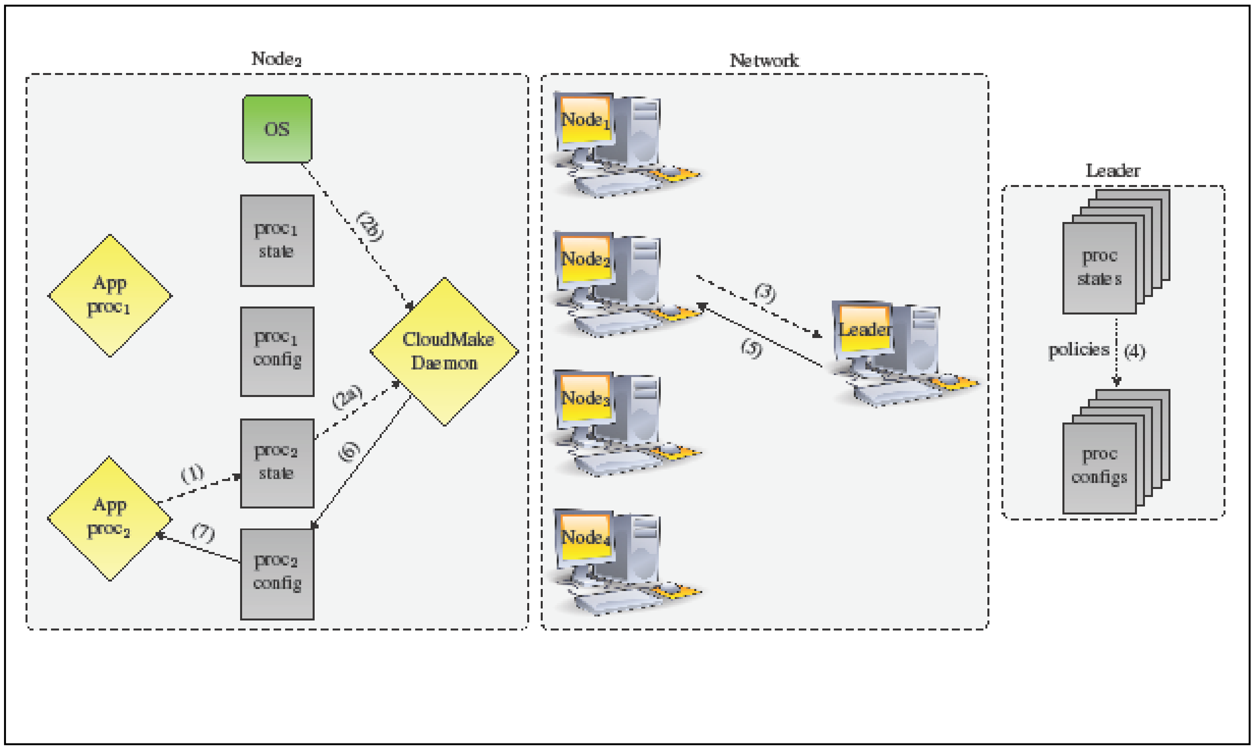

Figure 2: Using CloudMake to manage GridCloud

A final component of GridCloud is the Freeze Frame File System, discussed previously. In this role, FFFS captures and archives data received from the PMU and PDC data streams, as well as other incoming updates such as changes to the network model, updates to key parameters, etc. This data is recorded into files, and applications can thus either tap into the live streams and run in real-time, or can run on the archived data in near-real time or even long after an event. The temporal read features of GridCloud's FFFS allow the users to summon up any past network state they desire, much like a database query but without forcing data into a database schema, cleaning data in ways that might hide important inconsistencies evident in the raw streams, etc. This enables quick search of the past: has an event of this kind occurred previously? How often? What did we do last time we saw it? How well did that work? Over time, we expect that GridCloud will host many such solutions (in these very preliminary stages of use, we are still focused mostly on real time network state tracking: the continuous *state estimation* task.

Experiments, reported in [1], detail our findings. The replication scheme shown in figure 2 indeed allows us to mask network delays and disruptions, and replication internal to GridCloud lets us carry out state estimation redundantly to hide other problems such as overloaded cloud nodes or sudden migrations. We are able to track state even at national scale, with delays of hundreds of milliseconds between when sensors capture data and operators see it, even using a data center on the other side of the country as a backup for events occurring in the Northeast. And we can confirm strong consistency for the system as it runs. The ISO teams expect stringent security, and for these purposes, we were able to satisfy their requirements. Thus there is solid evidence that our MRC effort focused on the right problems, solved them, and produced a technology that the critical systems community can use.

5. CONCLUDING REMARKS

Our DARPA MRC effort set out to show that hybrid clouds blending private systems and sensors with commercial (government) cloud platforms could provide real-time guarantees, strong consistency, exceptional scalability, and the flexibility to leverage the low cost of the cloud without vendor lock-in and without putting sensitive data into the hands of cloud providers. We believe that our work was successful. We created powerful tools that are interesting in and of themselves, and then used them in an important real context, reflecting a major military priority as evidenced by the RADICS solicitation. Further, our work on leveraging RDMA technology is opening the door to huge speedups for data replication, which in turn could enable dramatic advances in the use of machine learning and computerized decision support for real-time cases such as to assist forward-deployed military personnel.

6. RECOMMENDATIONS

Looking to the future, we would urge AFRL and DARPA to think about the general power and potential evidenced by our success not merely solving these foundational problems, but also in applying the solutions to create GridCloud for the bulk electric power community. We have a proposal pending in the RADICS program and should not lobby for it here. But we do wish to end on the observation that there are many settings in which a real-time cloud, hardened with military use cases in mind, could transform our options both in the services today (we would point to the AF concept of "Distributed Operations" as an example: our work offers a possible substrate for supporting this important idea), and in the government and private sector, where the need for robust, secure, consistent systems with powerful real-time guarantees is already apparent and will surely grow over time.

More broadly, we'll simply reiterate a point mentioned in the discussion above. Recent success with machine learning has in some ways distracted from the need to reliably and security get the data to the machine learning solutions, to run them reliably and security, and then to get decision support back to the forward-deployed combatant or to other front line system elements. We need to invest and carry out research to create these kinds of ultra-fast, ultra-reliable infrastructures, because without them, even the most amazing advancing is machine intelligence will be inapplicable to the settings where we would wish to leverage them. Our success with the power grid project should thus be seen as a hint of what can be accomplished in a dozen such settings: in Air Force distributed operations systems, for example, or in decision support for pilots in our cutting-edge aircraft. The game isn't purely one of machine learning: getting the data to the AI system, and taking actions with minimal delay, is at least as important an objective! Our DARPA MRC successes point to a much larger opportunity.

7. REFERENCES

1. Dave Anderson, Dave Bakken, Ken Birman, Anjan Bose, Carl Hauser, Theo Gkountouvas, Robbert van Renesse, and Weijia Song. Hosting the Smart Grid: The GridCloud Platform for Monitoring and Controlling the Bulk Power Grid. Submitted to IEEE Power and Energy Magazine, December 2015.

2. Hussam Abu-Libdeh, Robbert van Renesse, Ymir Vigfusson. Leveraging Sharding in the Design of Scalable Replication Protocols. In *Proceedings of the 4th Annual Symposium on Cloud Computing (SoCC)*, Santa Clara, CA. October 2013.

3. Kenneth P. Birman. Guide to Reliable Distributed Systems. Building High-Assurance Applications and Cloud-Hosted Services (Texts in Computer Science). Springer-Verlag London. 2012.

4. Ken Birman. The Roles of Formal Methods in Building High Assurance Cloud Computing Platforms. Presented at T*he 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2012)*, October 2012.

5. Ken Birman. "Vsync: Data Replication for Cloud Computing". Vsync.codeplex.com (includes papers, documentation, training videos, source code), last modified Jan 30, 2016. Accessed Feb 2, 2016.

6. Ken Birman. Reflections on the History of Operating Systems Research in Fault Tolerance. *SOSP History Day*. October 2015.

7. Ken Birman. "GridControl: A Software Platform to Support the Smart Grid". http://www.cs.cornell.edu/projects/gridcontrol/, last modified Jan 30, 2016. Accessed February 16, 2016.

8. Jonathon Behrens, Ken Birman, Sagar Jha, and Edward Tremel. RDMC: A Reliable RDMA Multicast Protocol. Submitted to *Middleware 2016*.

9. K. P. Birman, Q. Huang, D. Freedman, P. Dowell. Overcoming CAP with Consistent Soft-State Replication. *IEEE Computer Magazine* (special issue on "The Growing Impact of the CAP Theorem"). Volume 45. pp. 50-58. February 2012.

10. Ken Birman, Márk Jelasity, Robert Kleinberg, Edward Tremel. Building a Secure and Privacy-Preserving Smart Grid. ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts. Vol. 49(1), January 2015.

11. Eduardo Bezerra, Fernando Pedone, and Robbert van Renesse. Scalable State-Machine Replication. *The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)*, Atlanta, GA, June 2014.

12. Ken Birman and Heesung Sohn. Hosting Dynamic Data in the Cloud with Isis2 and the Ida DHT. In *Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems (TRIOS)*, November 2013.

13. Ittay Eyal, Ken Birman, Idit Keidar, and Robbert van Renesse. Ordering Transactions with Prediction in Distributed Object Stores. *LADIS 2013: 7th Workshop on Large-Scale Distributed Systems and Middleware*, November 2013.

14. Ittay Eyal, Ken Birman, and Robbert van Renesse. Controlled Transactional Consistency for Web Caching. arXiv:1409.8324v1 Technical Report, Cornell University. October 2014.

15. Ittay Eyal, Ken Birman, Robbert van Renesse. Managed Transactional Consistency for Web Caching, *35th IEEE international conference on Distributed Computing Systems (ICDCS'15)*, June 2015.

16. Ittay Eyal, Ken Birman, Robbert van Renesse. Cache Serializability: Reducing Inconsistency in Edge Transactions. Intl. Conference on Distributed Computing Systems (ICDCS), IEEE 35th International Conference on, June 29 2015–July 2 2015.

17. Ittay Eyal and Emin Gun Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. *The Eighteenth International Conference on Financial Cryptography and Data Security (FC'14)*, March 2014.

18. Thoshitha Gamage, David Anderson, David Bakken, Kenneth Birman, Anjan Bose, Carl Hauser, Ketan Maheshwri, and Robbert van Renesse. Mission-Critical Cloud Computing for Next-Generation Power Applications. In Smart Grids: Clouds, Communications, Open Source, and Automation. Editors David Bakken and Kris Iniewski. CRC Press. ISBN 9781482206111. 2014.

19. Theodoros Gkountouvas, Kenneth Birman, and David Anderson. DMake: A Tool for Monitoring and Configuring Distributed Systems. Submitted to *Middleware 2014*.

20. Haoyan Geng and Robbert van Renesse. Sprinkler -- Reliable Broadcast for Geographically Dispersed Datacenters. In *Proceedings ACM/IFIP/USENIX 14th International Middleware Conference*, Beijing, China, pp.247-266. December 2013.

21. Qi Huang, Helga Gudmundsdottir, Ymir Vigfusson, Daniel Freedman, Ken Birman, and Robbert van Renesse. Characterizing Load Imbalance in Real-World Networked Caches. In *Proceedings of the Thirteenth ACM Workshop on Hot Topics in Networks (HotNets '14)*, Los Angeles, CA, October 2014.

22. Qi Huang, Ken Birman, Robbert van Renesse, Wyatt Lloyd, Sanjeev Kumar, Harry C. Li. An Analysis of Facebook Photo Caching. *SOSP 2013: The 24th ACM Symposium on Operating Systems Principles*, November 2013.

23. Qin Jia, Zhiming Shen, Weijia Song, Robbert van Renesse, and Hakim Weatherspoon. Supercloud: Opportunities and Challenges. *ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts.* Vol. 49(1), pp.137-141, January 2015.

24. Haavard Johansen, Robbert van Renesse, Ymir Vigfusson, and Dag Johansen. Fireflies: Secure and Scalable Membership and Gossip Service. *ACM Transactions on Computer Systems*, Vol. 33(2), June 2015.

25. Parisa Jalili Marandi, Samuel Benz, Fernando Pedone, Ken Birman. Practical Experience Report: The Performance of Paxos in the Cloud. *The 33rd IEEE Symposium on Reliable Distributed Systems (SRDS 2014),* Nara, Japan, October 2014.

26. Ketan Maheshwari, Marcus Lim, Lydia Wan, Ken Birman, Robbert van Renesse. Toward a reliable, secure and fault tolerant smart grid state estimation in the cloud. Proceedings of *IEEE/PES Innovative Smart Gird Technologies (ISGT)*. pp 1-6. February 2013.

27. Lonnie Princehouse, Rakesh Chenchu, Zhefu Jiang, Kenneth P. Birman, Nate Foster, and Robert Soule. MiCA: A Compositional Architecture for Gossip Protocols. *The 28th European Conference on Object-Oriented Programming (ECOOP 2014)*. Uppsala, Sweden, July 2014.

28. Lonnie Princehouse, Ken Birman, and Nate Foster. Compositional Gossip Protocols for Infrastructure Management. *LADIS 2013: 7th Workshop on Large-Scale Distributed Systems and Middleware*, November 2013.

29. Robert Soule, Ken Birman, Nate Foster. Software Defined Networks and Gossip Protocols. The 8th Workshop on Large-Scale Distributed Systems and Middleware, LADIS 2014. Cambridge, UK. October 2014.

30. Robert Surton, Ken Birman, and Robbert van Renesse. Application-Driven TCP Recovery. In *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Budapest, Hungary, June 2013.

31. Weijia Song, Theodoros Gkountouvas, Ken Birman, Qi Chen, Zhen Xiao. FFFS: The Freeze-Frame File System. Submitted to FAST 2016.

32. Nicolas Schiper, Vincent Rahli, Robbert van Renesse, Mark Bickford, and Robert L. Constable. ShadowDB: A Replicated Database on a Synthesized Consensus Core. In *Proceedings of the Eighth Workshop on Hot Topics in System Dependability (HotDep),* Hollywood, CA. October 2012.

33. Edward Tremel, Ken Birman, Robert Kleinberg, Mark Jelasity. A Private Framework for Distributed Computation. The 8th Workshop on Large-Scale Distributed Systems and Middleware, LADIS 2014. Cambridge, UK. October 2014.

34. Zhiyuan Teo, Ken Birman, and Robbert van Renesse. Lessons in OpenFlow hardware and controller symbiosis. Submitted to NSDI 2016.

35. Linpeng Tang, Qi Huang, Wyatt Lloyd, Sanjeev Kumar, and Kai Li. RIPQ: Advanced Photo Caching on Flash for Facebook. *13th USENIX Conference on File and Storage Technologies (FAST '15)*. Santa Clara, CA. February 2015.

36. Zhiyuan Teo, Vera Kutsenko, Ken Birman, and Robbert van Renesse. IronStack: performance, stability and security for power grid data networks. *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).* pp.792-797, Atlanta, GA, June 2014.

37. Ymir Vigfusson, Ken Birman, Daniel A. Freedman, Qi Huang, Kristjan Jonsson, Gunnar Sigurbjornsson. Live Network Streaming with Utilities and Cost. *LADIS'12*, Madeira, Portugal, July 16–18, 2012**.**

38. Robbert van Renesse, Haavard Johansen, Nihar Naigaonkar, and Dag Johansen. Secure Abstraction with Code Capabilities. In *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*. Belfast, Northern Ireland. February 2013.

39. Robbert van Renesse and Deniz Altinbuken. Paxos Made Moderately Complex. *ACM Computing Surveys*, Vol. 47(3). April 2015.

40. Robbert van Renesse, Chi Ho, Nicolas Schiper. Byzantine Chain Replication. In *Proceedings of 16th International Conference on Principles of Distributed Systems (OPODIS 2012)*, Rome, Italy, December 2012.

41. Robbert van Renesse, Nicolas Schiper, and Fred B. Schneider. Vive la Difference: Paxos vs. Viewstamped Replication vs. Zab. *IEEE Transactions on Dependable and Secure Computing*. 2014.

42. Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. The Xen-Blanket: Virtualize Once, Run Everywhere. In *Proceedings of the 7th ACM European Conference on Computer Systems (Eurosys),* April 2012.

43. Dan Williams, Hani Jamjoom, and Hakim Weatherspoon. Software Defining System Devices with the "Banana" Double-Split Driver Model. *The 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 2014)*. Philadelphia, PA, June 2014.

44. Han Wang, Ki Suh Lee, Erluo Li, Chiun Lin Lim, Ao Tang, and Hakim Weatherspoon. Timing is Everything: Accurate, Minimum Overhead, Available Bandwidth Estimation in High-speed Wired Networks. *14th ACM SIGCOMM Internet Measurement Conference (IMC),* Vancouver, Canada, November 2014.

45. Dan Williams, Hani Jamjoom, and Hakim Weatherpoon. Plug into the Supercloud, *IEEE Internet Computing*, Vol 17(2), pp.28-34, March 2013.

46. Dan Williams, Eslam Elnikety, Mohamed Eldehiry, Hani Jamjoom, Hai Huang, and Hakim Weatherspoon. Unshackle the Cloud! In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, June 2011.

47. H.Weatherspoon, L. Ganesh, T. Marian, M. Balakrishnan, and K. Birman. Smoke and mirrors: Shadowing files at a geographically remote location without loss of performance. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, February 2009.

48. K. Birman and T. Joseph. Exploiting Virtual Synchrony in Distributed Systems. *11th ACM Symposium on Operating Systems Principles (SOSP)*, December 1987.

49. Jonathon Behrens. “RDMC”. http://rdmc.codeplex.com/. Last modified Sep 24, 2015. Accessed Feb 2, 2016.

50. David Patterson, Garth Gibson, and Randy Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceeding of the 1988 ACM SIGMOD international conference on Management of Data*. Vol 17(3), pp.109-116, June 1988.

51. Douglas E. Comer and David L. Stevens. Internetworking with TCP/IP, Vol. III: Client-Server Programming and Applications, Linux/Posix Sockets Version. Tankobon Publishing Company. Sep 21, 2000.

ADDITIONAL SOURCES

JB14

Mark Jelasity and Kenneth P. Birman. Distributional Differential Privacy for Large-Scale Smart Metering. *The 2nd ACM Workshop on Information Hiding and Multimedia Security (IH & MMSEC 2014),* Salzburg, Austria, June 2014.

BMvR12

Ken Birman, Dahlia Malkhi, Robbert van Renesse. Virtually Synchronous Methodology for Dynamic Service Replication. In *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services-Appendix A (Chapter 22*) Springer-Verlag London, pp635-671, February 2012.

E+16

Ittay Eyal, Adam Efe Gencer, Emin Gun Sirer, and Robbert van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16),* March 2016.

L+14

Chiun Lin Lim, Ki Suh Lee, Han Wang, Hakim Weatherspoon, and Ao Tan. Packet clustering introduced by routers: Modeling, analysis and experiments. *48th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2014.

LWW14
Ki Suh Lee, Han Wang, and Hakim Weatherspoon. PHY Covert Channels: Can you see the Idles? *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14),* Seattle, WA, April 2014.

RS+12
Vincent Rahli, Nicolas Schiper, Robbert Van Renesse, Mark Bickford, and Robert L. Constable. A Diversified and Correct-by-Construction Broadcast Service. In *Proceedings of the 2nd International Workshop on Rigorous Protocol Engineering (WRiPE).* Austin, TX, October 2012.

SB+13
Ji-Yong Shin, Mahesh Balakrishnan, Tudor Marian, and Hakim Weatherspoon. Gecko: Contention-Oblivious Disk Arrays for Cloud Storage. *The 11th USENIX conference on File and Storage Technologies (FAST).* February 2013.

SPvR14
Nicolas Schiper, Fernando Pedone, and Robbert van Renesse. The Energy Efficiency of Database Replication Protocols. *The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014).* Atlanta, GA. June 2014.

WJ+13a
Dan Williams, Hani Jamjoom, Zhefu Jiang, and Hakim Weatherspoon. SoNIC: Precise Realtime Software Access and Control of Wired Networks. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation,* April 2013.

WJ+13b
Dan Williams, Zhefu Jiang, Hani Jamjoom, and Hakim Weatherspoon. Virtual Wires for Live Migrating Virtual Networks across Clouds. *IBM Technical Report* RC25378, April 2013.

8. LIST OF ACRONYMS

AFRL	Air Force Research Lab
CIO	Chief Information Officer
CRASH	Clean-slate design of Resilient, Adaptive, Secure Hosts
CTO	Chief Technology Officer
DARPA	Defense Advanced Research Projects Agency
DHT	Distributed Hash Table
DMA	Direct Memory Access
DOE	Department of Energy

EC2	Elastic Compute Cloud
FFFS	Freeze-Frame File System
fVMs	first-layer Virtual Machines
HACC	High Assurance Cloud Computing
HP	Hewlett-Packard
HPC	High Performance Computing
IBM	International Business Machines
ISO NE	Independent System Operator New England
LLNL	Lawrence Livermore National Laboratory
MRC	Mission Resilient Cloud
NSDI	Networked Systems Design and Implementation
NY ISO	New York Independent Systems Operator
NYPA	New York Power Authority
RADICS BAA	Rapid Attack Detection, Isolation and Characterization Systems Broad Area Announcement
RAID	Redundant Array of Independent Disks
RDMA	Remote Direct Memory Access
RDMC	Reliable DMA Multicast
SDN	Software Defined Networks
SST	Shared State Table
sVMs	second-layer Virtual Machines
TCP	Transmission Control Protocol
VMM	Virtual Machine Monitor